

FIG. 1

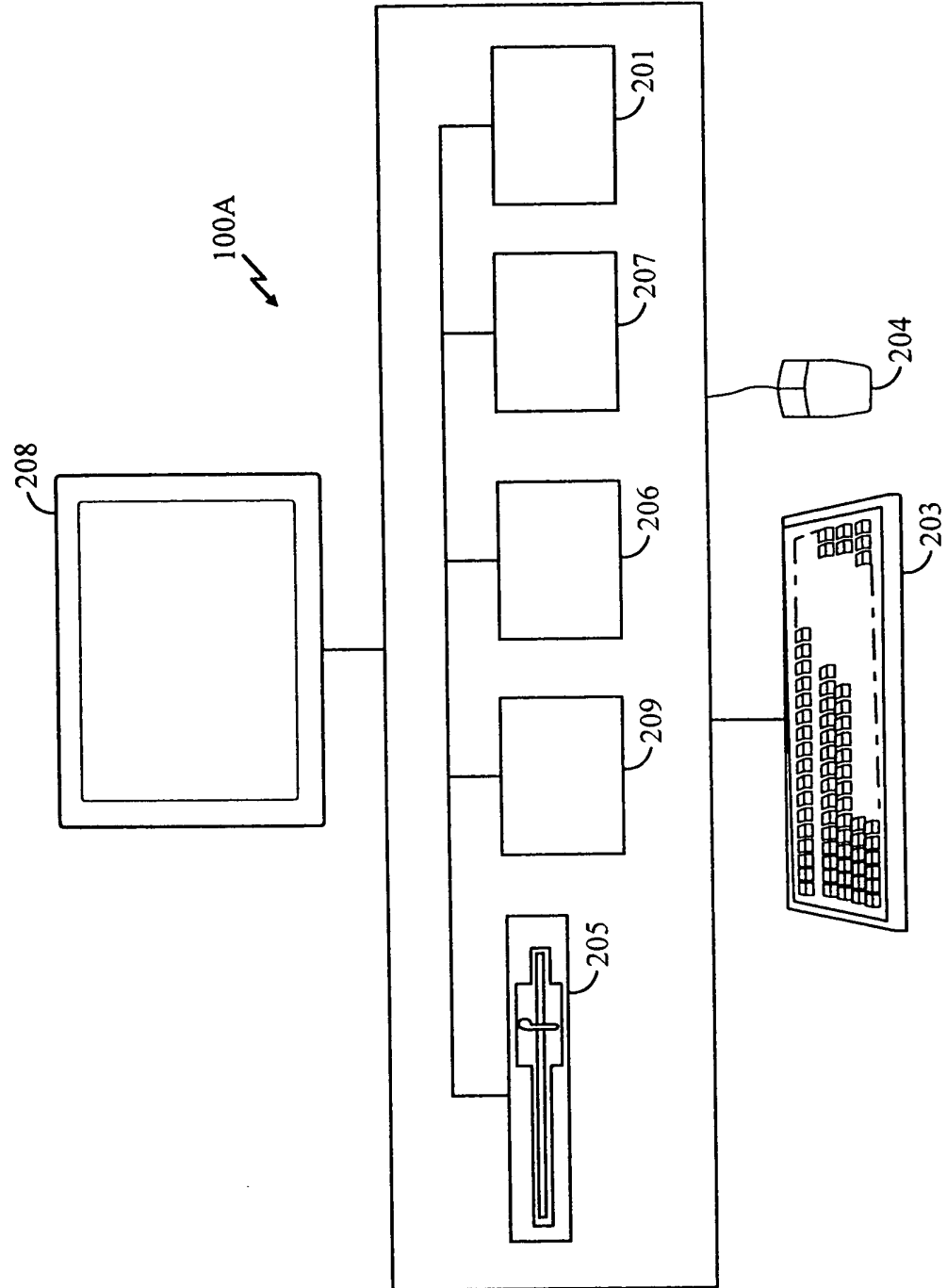


FIG. 2

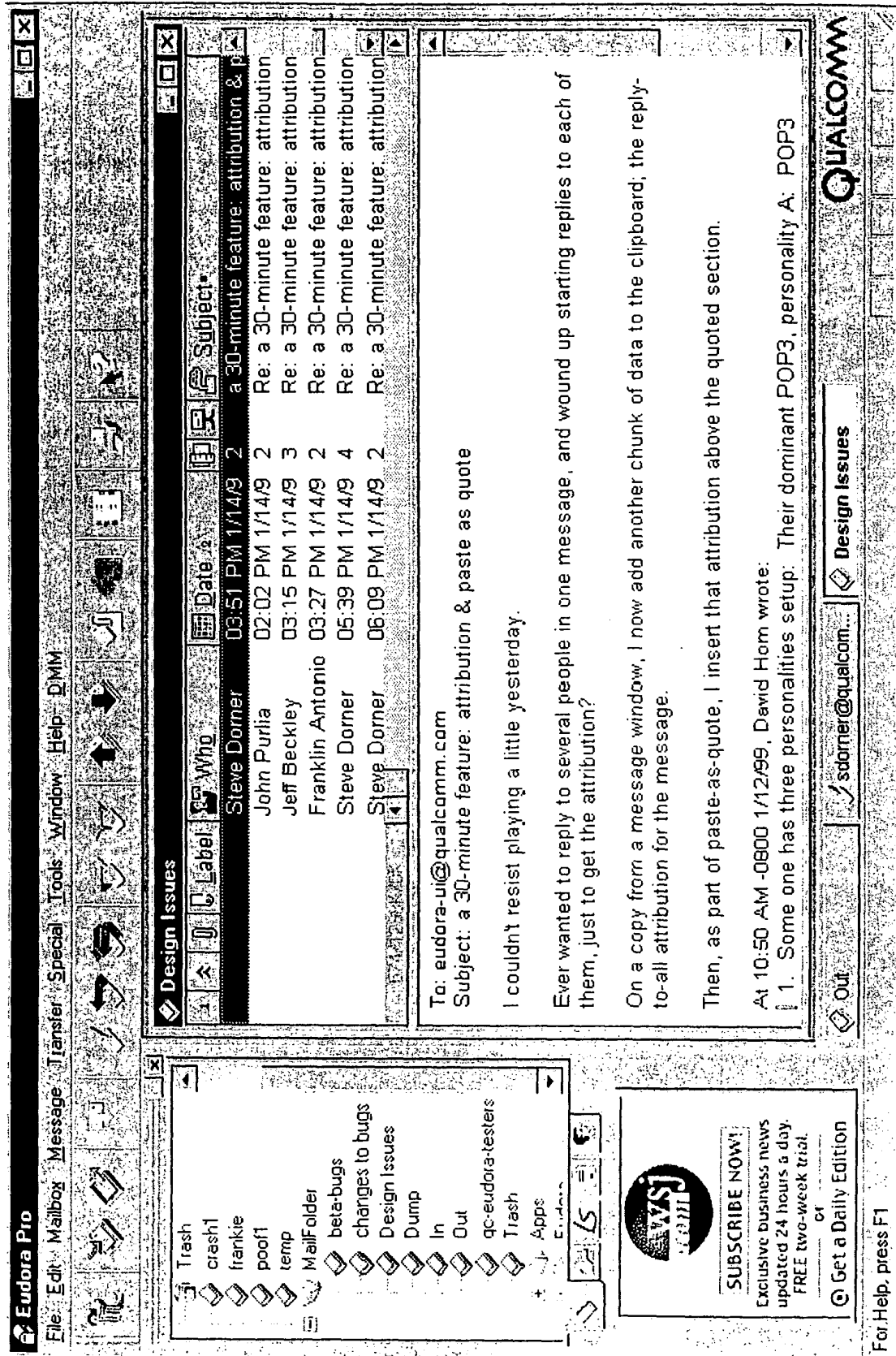


FIG. 3A

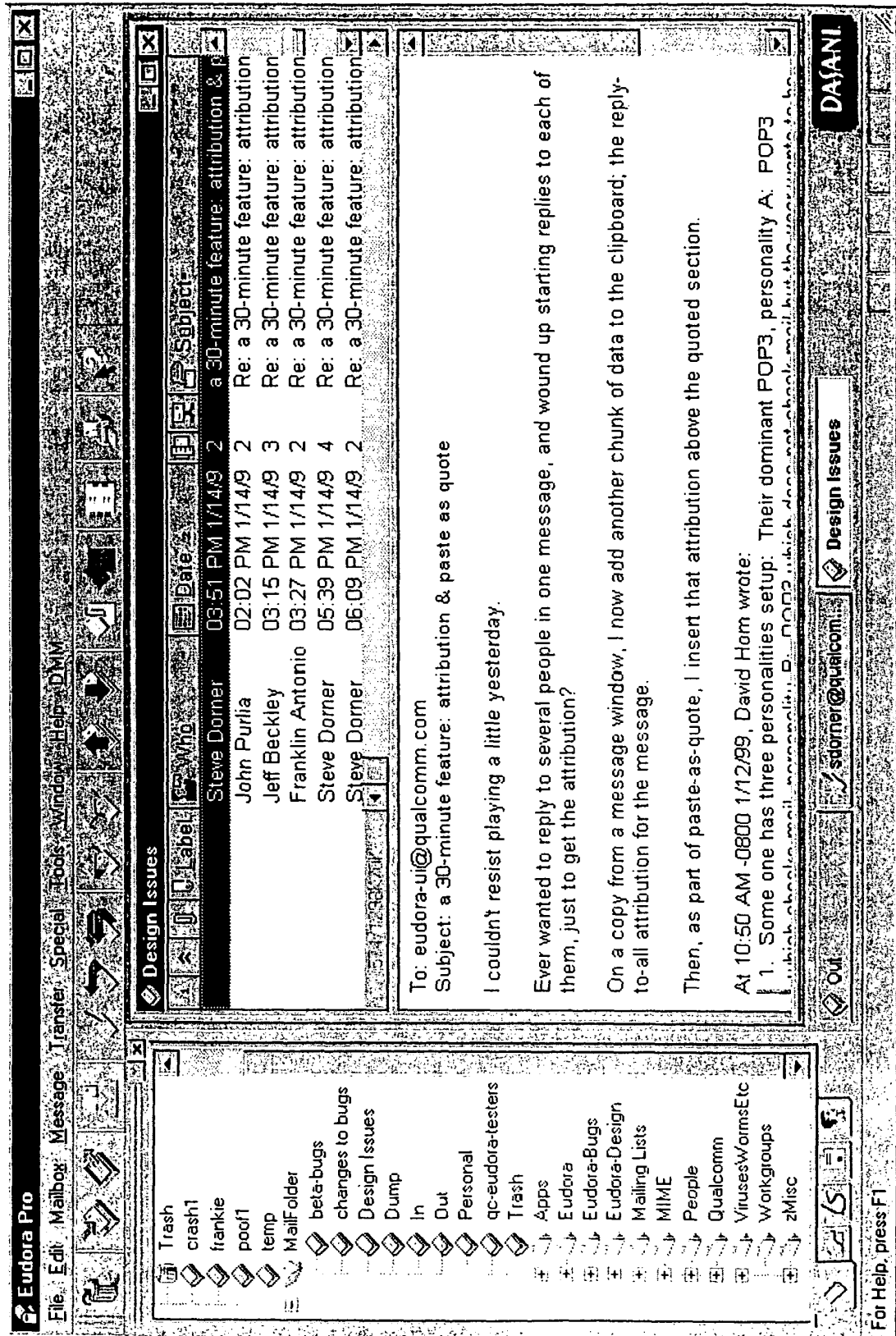


FIG. 3B

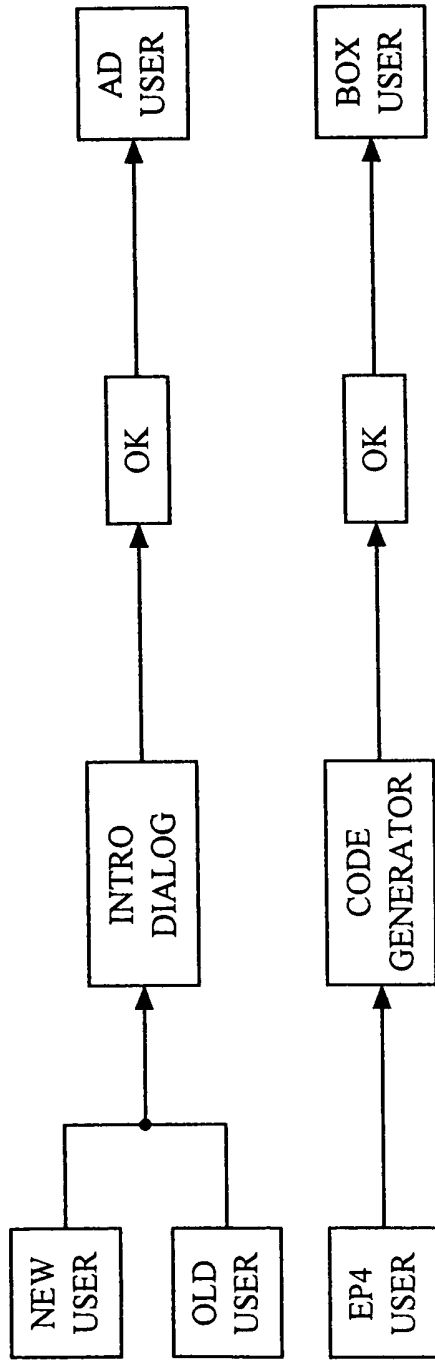


FIG. 4A

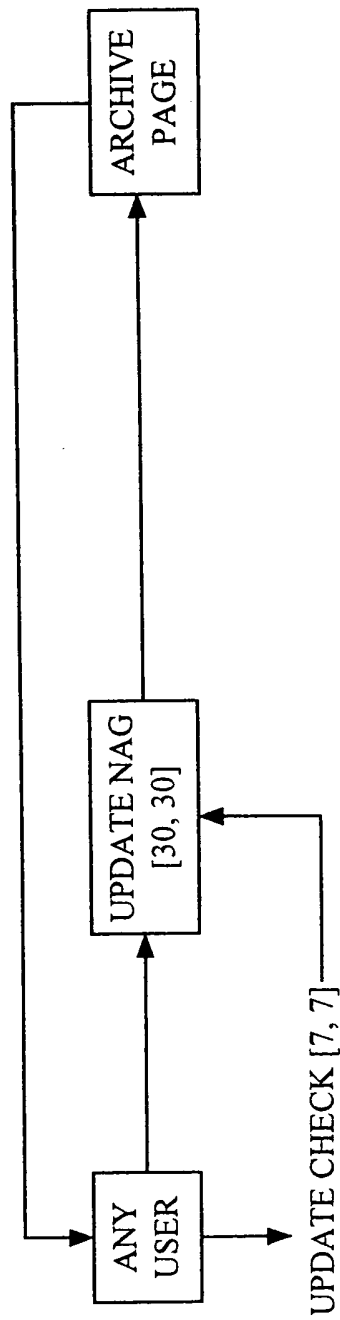


FIG. 7A

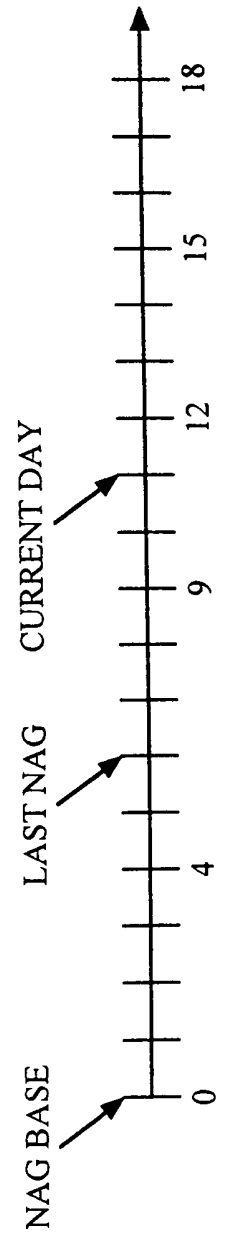


FIG. 11

WELCOME TO EUDORA!

EUDORA IS NOW LICENSED IN THREE WAYS; SPONSORED MODE, PAID MODE, AND LIGHT MODE. UNLESS YOU CHANGE MODES, EUDORA WILL RUN IN SPONSORED MODE, MEANING IT WILL DISPLAY ADS.

WE HAVE DONE OUR BEST TO PRESENT THE ADS IN A WAY THAT RESPECTS THE WORK YOU DO IN EMAIL. BY ALLOWING EUDORA TO DISPLAY ADS, YOU GET THE FULL POWER OF EUDORA FOR FREE AND WE CAN STILL PAY OUR BILLS.

IF YOU DECIDE THE ADS ARE NOT FOR YOU, YOU CAN CHANGE MODES. PAID MODE SHOWS NO ADS. CURRENT EUDORA PRO 4.X USERS WILL BE ABLE TO UPGRADE TO PAID MODE FOR FREE. OTHER USERS WILL BE ABLE TO PAY A LICENSE FEE TO GO TO PAID MODE. AT THIS STAGE IN TESTING, THE MACHINERY FOR PAID MODE IS NOT FULLY TESTED, AND PAID MODE IS UNAVAILABLE. LIGHT MODE ALSO SHOWS NO ADS, BUT HAS MANY FEWER FEATURES.

TO SWITCH FORMS OF EUDORA, PLEASE USE THE "PAYMENT & REGISTRATION" ITEM IN THE HELP MENU. TO LEARN MORE ABOUT THE THREE MODES, CLICK ON THE "TELL ME MORE" BUTTON BELOW.

TELL ME MORE

OK

FIG. 4B

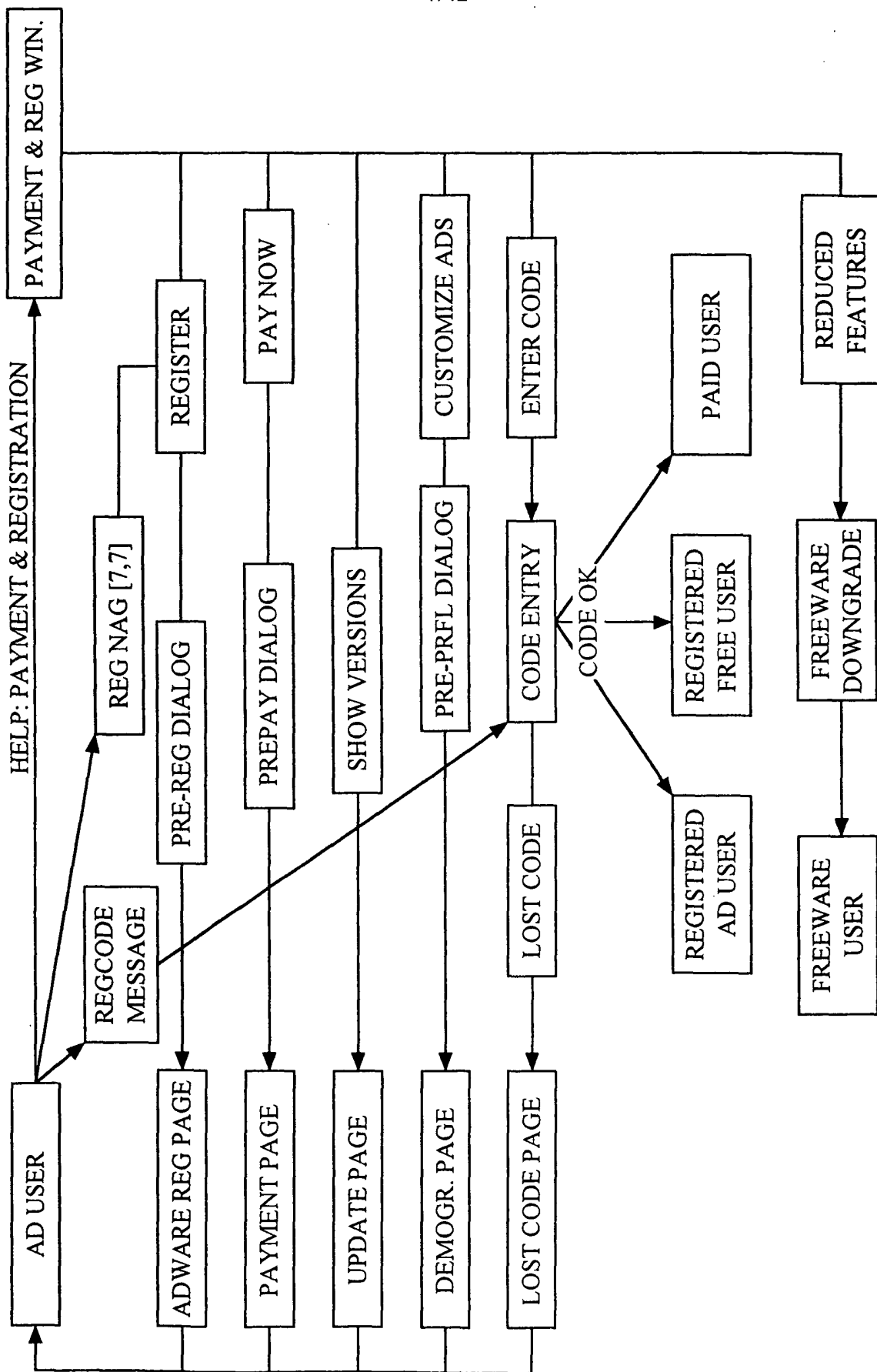


FIG. 5A

☐
☐

PAYMENT & REGISTRATION

WHICH EUDORA IS RIGHT FOR YOU?

SPONSERD MODE
(FREE, WITH ADS)

PAID MODE
(COST MONEY,
NO ADS)

LIGHT MODE
(FREE, FEWER
FEATURES)

KEEPING CURRENT

REGISTER
WITH US

CUSTOMIZE
THE ADS
YOU SEE

FIND THE
LATEST UPDATE
TO EUDORA

YOUR REGISTRATION INFORMATION

< NO REGISTRATION NAME >

< NO REGISTRATION CODE >

CHANGE
YOUR
REGISTRATION

TAKE ME TO THE TO THE EUDORA WEB SITE FOR MORE INFORMATION

FIG. 5B

FIG. 5C

— THANKS FOR CHOOSING TO REGISTER EUDORA! —

YOU'LL NEXT BE WALKED THROUGH A FEW QUICK STEPS, AS DESCRIBED BELOW, BEFORE REGISTRATION IS COMPLETE:

- EUDORA WILL OPEN YOUR WEB BROWSER AND TAKE YOU TO OUR REGISTRATION PAGE
- WE'LL FILL IN SOME SIMPLE REGISTRATION INFORMATION ON THE WEB SITE
- WE'LL THEN EMAIL A EUDORA REGISTRATION CODE BACK TO YOU
- THE NEXT TIME YOU CHECK MAIL, EUDORA WILL AUTOMATICALLY RECOGNIZE THIS CODE AND DISPLAY A DIALOG BOX INVITING YOU TO CONFIRM YOUR REGISTRATION INFORMATION
- TA DA! YOU'LL THEN BECOME A REGISTERED USER OF EUDORA... THANKS!

CANCEL CONTINUE

FIG. 5D

— THANKS FOR CHOOSING TO PURCHASE EUDORA! —

YOU'LL NEXT BE WALKED THROUGH A FEW QUICK STEPS, AS DESCRIBED BELOW, BEFORE YOUR PURCHASE IS COMPLETE:

- EUDORA WILL OPEN YOUR WEB BROWSER AND TAKE YOU TO OUR PAYMENT & REGISTRATION PAGE
- YOU'LL BE ASKED TO PROVIDE YOUR PAYMENT AND REGISTRATION INFORMATION ON THE WEB SITE
- WE'LL THEN EMAIL A EUDORA REGISTRATION CODE BACK TO YOU
- THE NEXT TIME YOU CHECK MAIL, EUDORA WILL AUTOMATICALLY RECOGNIZE THIS CODE AND DISPLAY A DIALOG BOX INVITING YOU TO CONFIRM YOUR REGISTRATION INFORMATION
- TA-DA! YOU'LL THEN BECOME A PAID MODE USER... CONGRATULATIONS!

CANCEL

CONTINUE

FIG. 5E

☐

☐

THANK YOU FOR YOUR REGISTRATION!

TO COMPLETE YOUR REGISTRATION, PLEASE ENTER THE NAME YOU REGISTERED UNDER AND YOUR REGISTRATION CODE BELOW.

THE EXACT NAME YOU REGISTERD UNDER:

FIRST NAME:

JOHN

LAST NAME:

MANYJARS

YOUR REGISTRATION CODE:

48925-89A2-B1149

I LOST THE CODE

CANCEL

OK

FIG. 5F

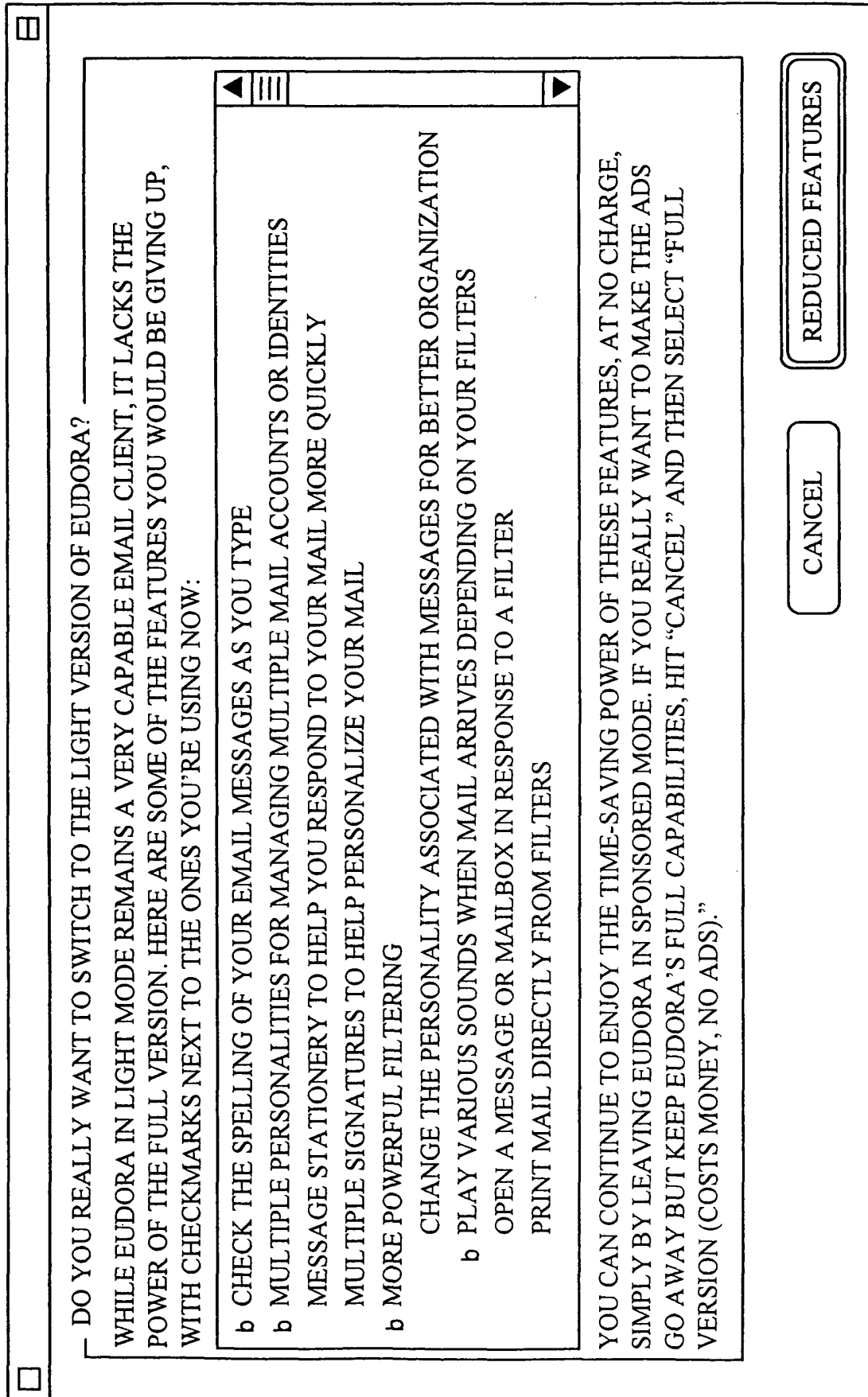


FIG. 5G

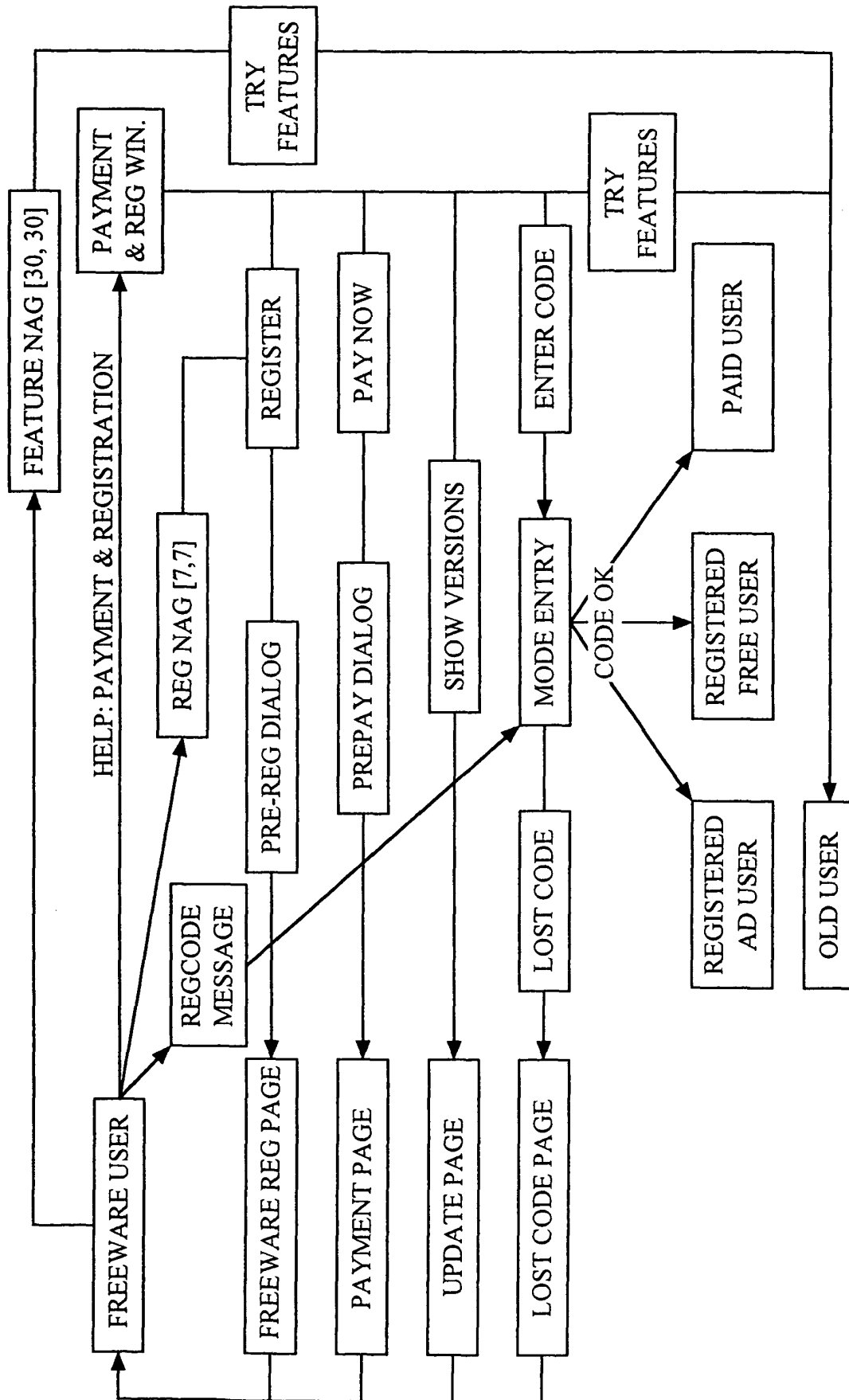


FIG. 6A

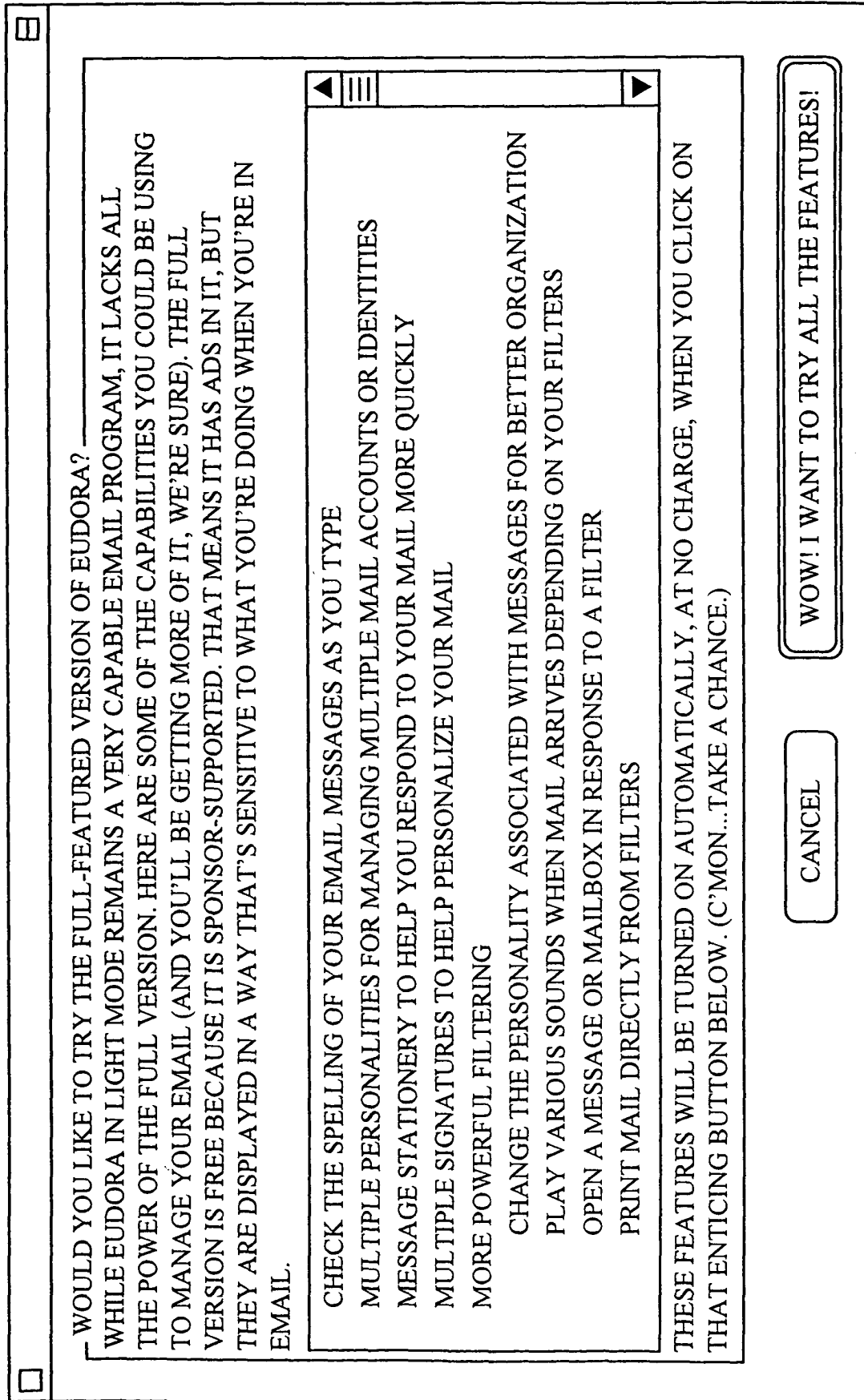


FIG. 6B

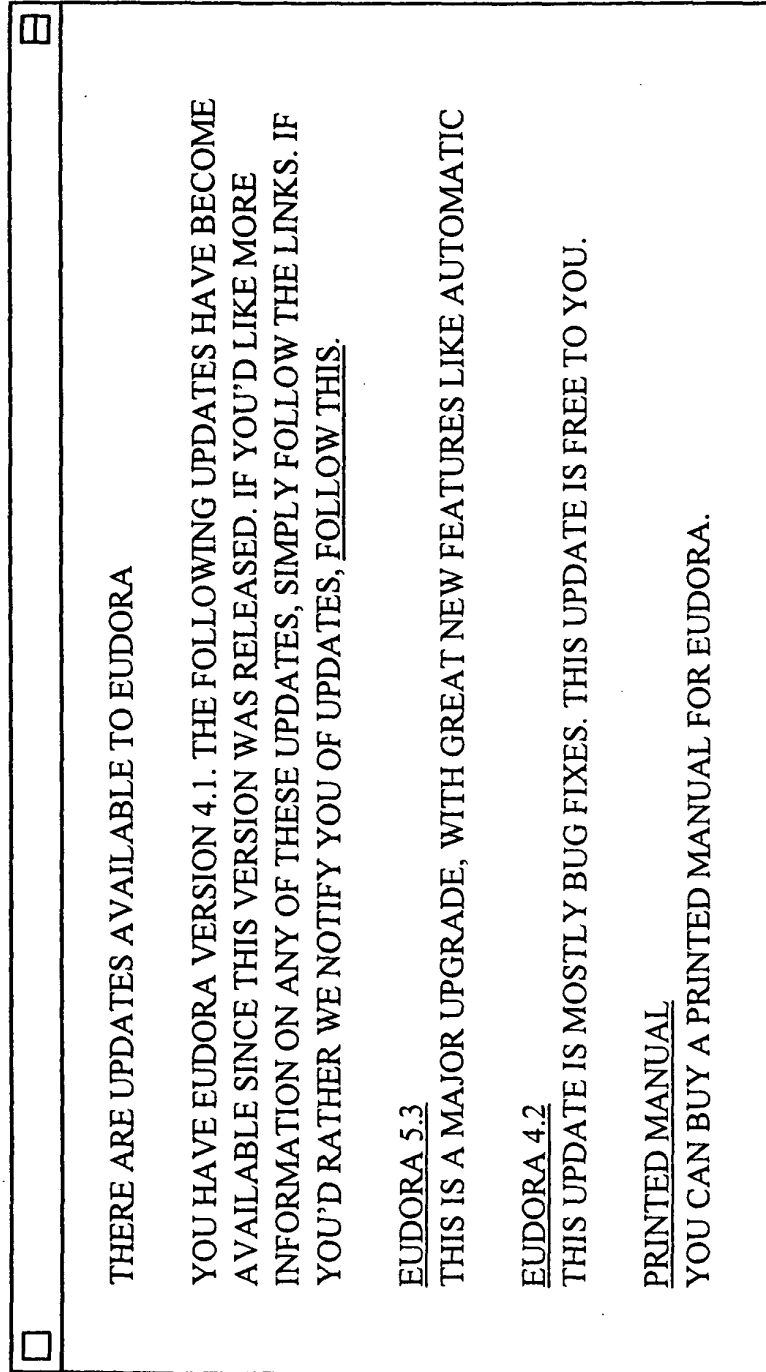


FIG. 7B

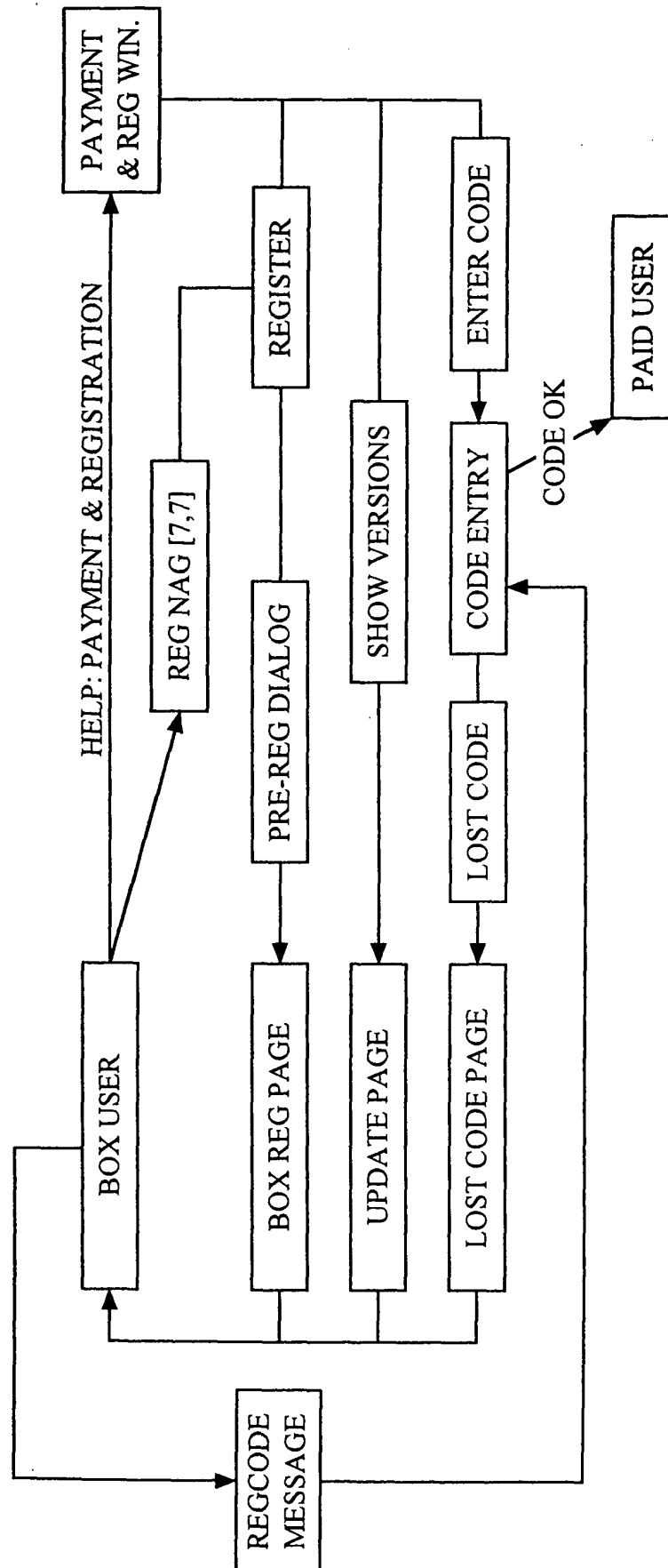


FIG. 8

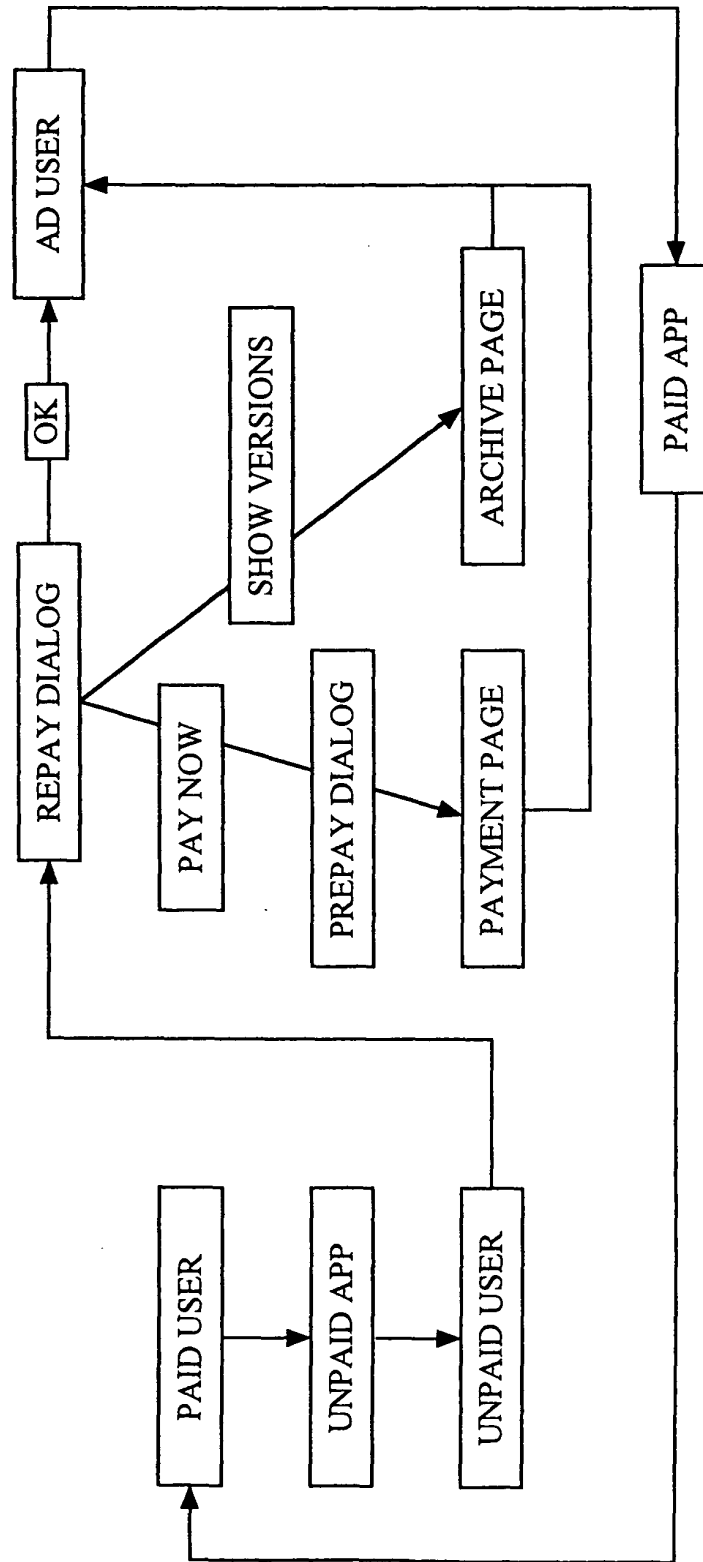


FIG. 9

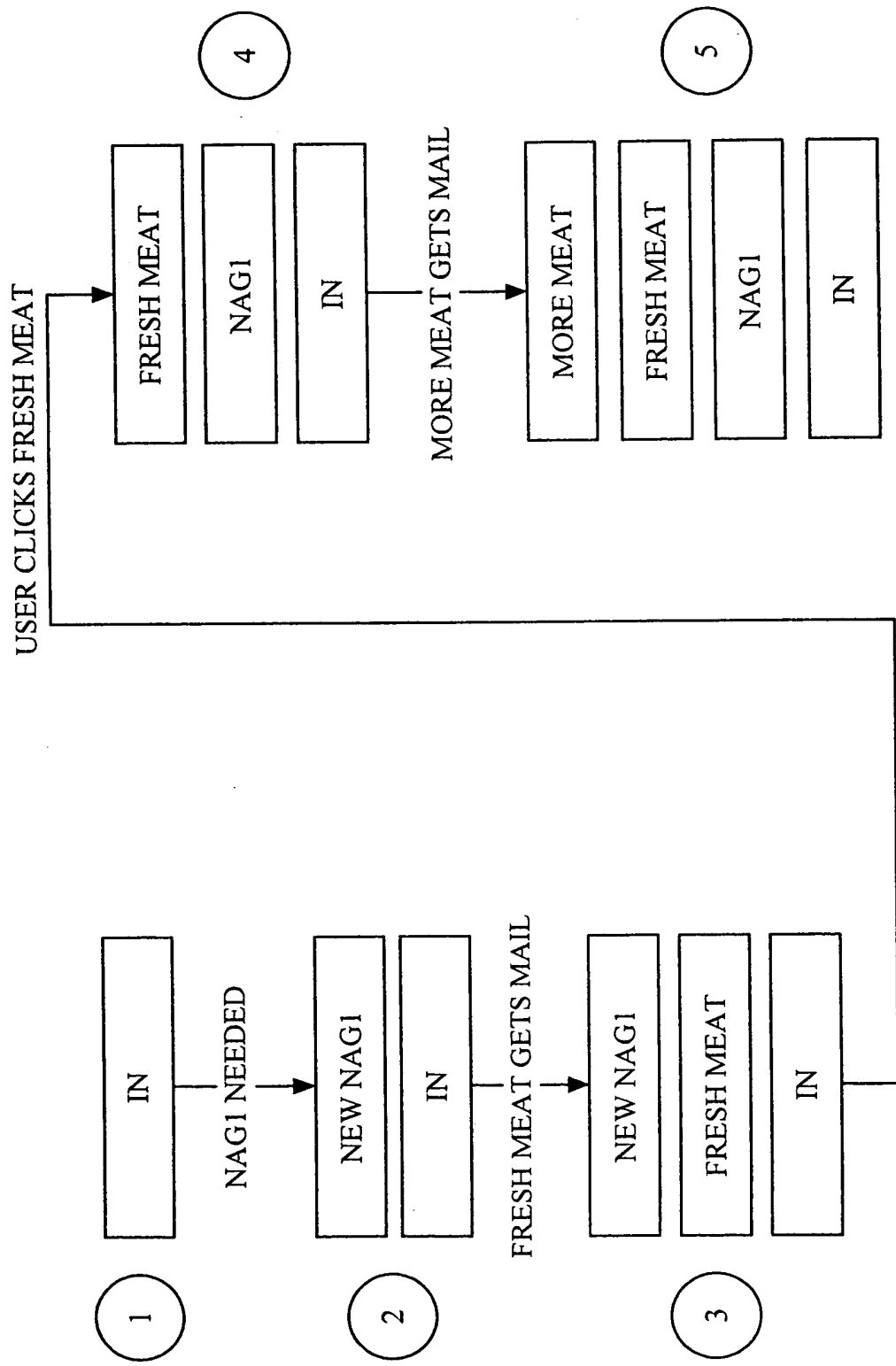


FIG. 10

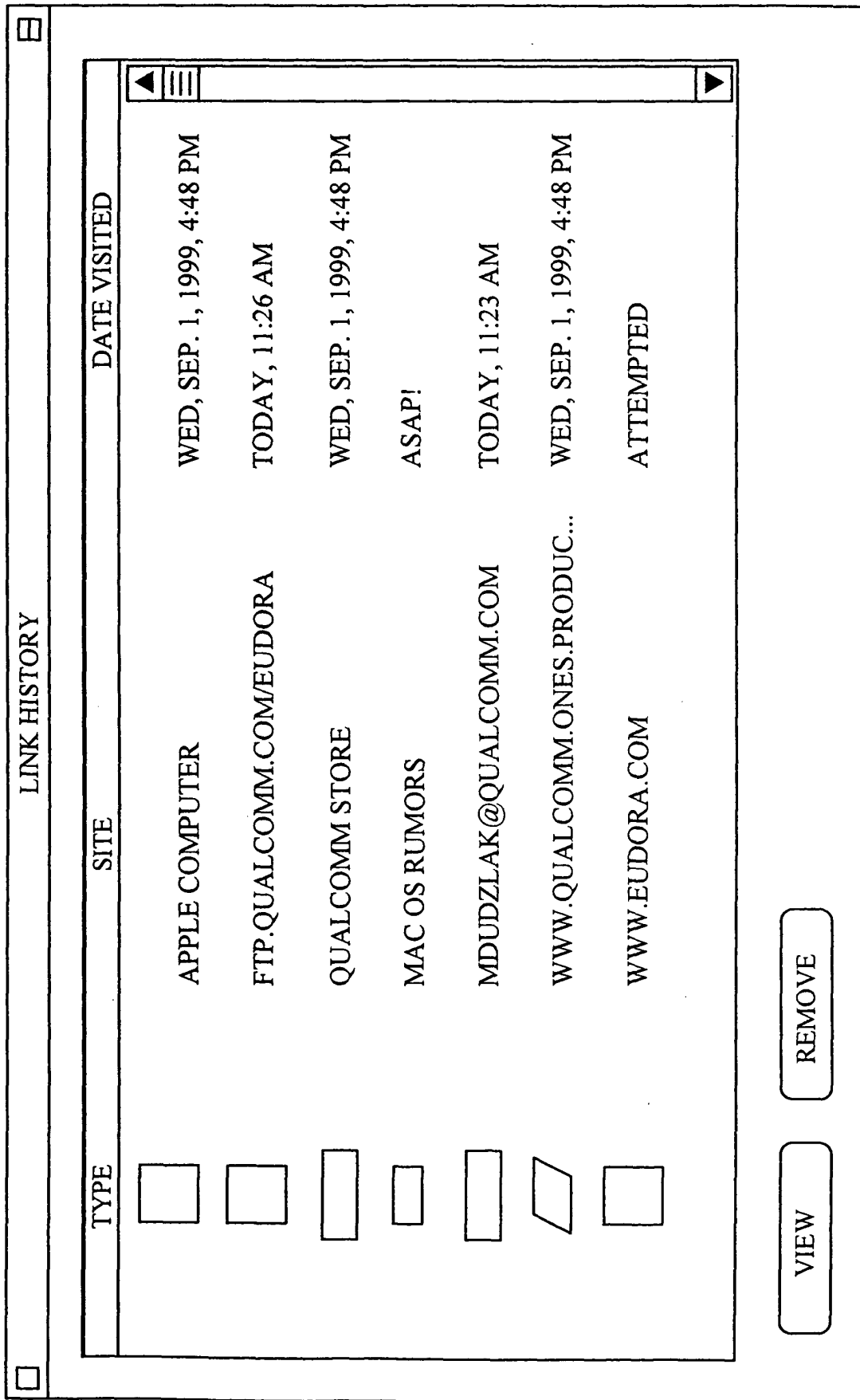


FIG. 12A

☐

YOU CAN'T GET THERE FROM HERE

YOU'RE NOT CONNECTED TO THE INTERNET NOW. HELP ME COPE.
CONNECT YOU AND VISIT THE SITE, RECORD A BOOKMARK FOR LATER
REMINDE YOU TO VISIT IT NEXT TIME YOU ARE CONNECTED.

CONNECT TO THE INTERNET AND VISIT THE SITE

BOOKMARK THIS SITE TO VISIT LATER

BOOKMARK THE SITE, AND REMIND YOU
YOU'RE CONNECTED TO THE INTERNET

☐ REMEMBER YOUR CHOICE FOR NEXT TIME

VISIT NOW

BOOKMARK

REMINDE ME

FIG. 12B

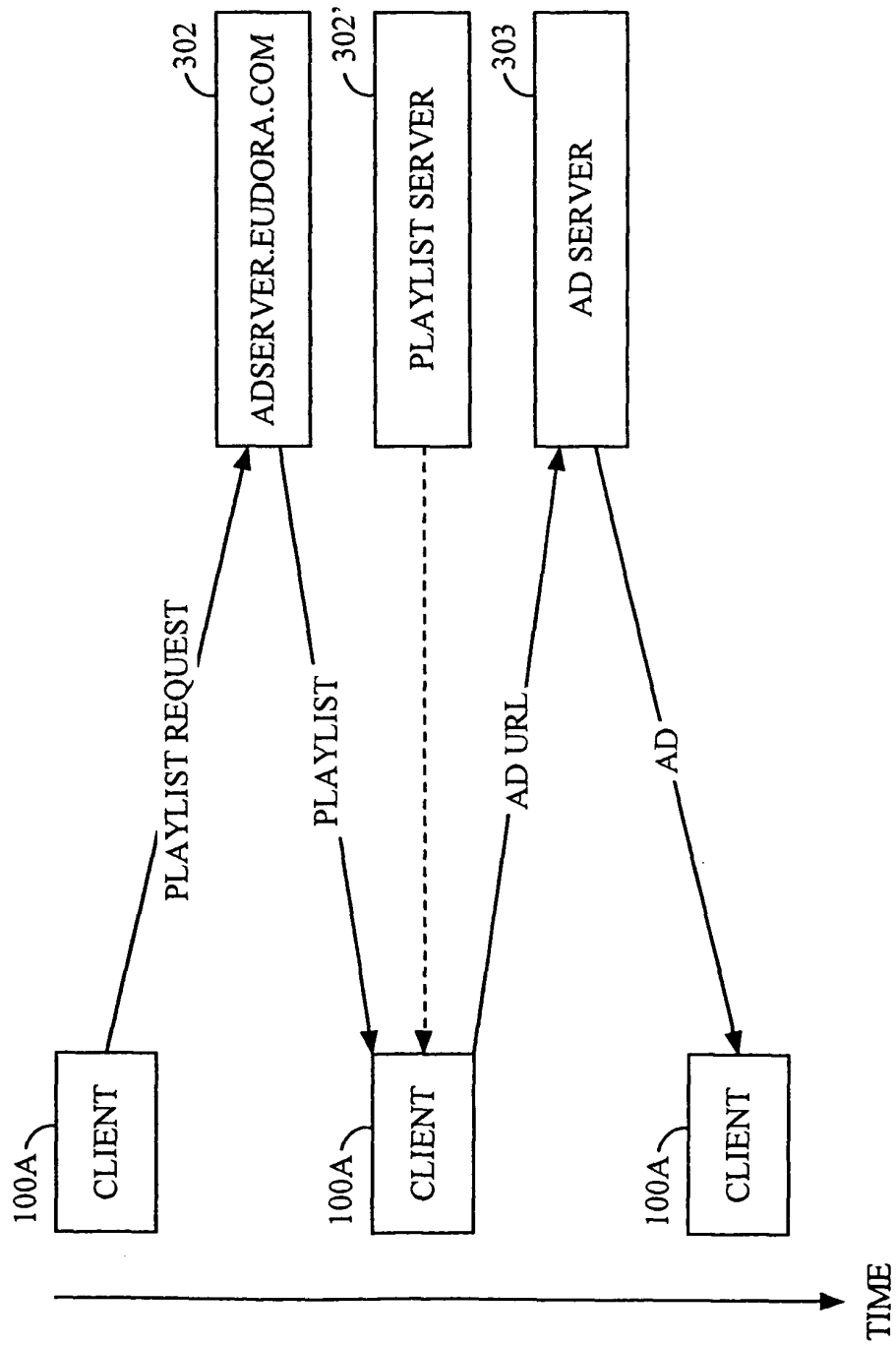


FIG. 14

```

////////////////////
// Main ad scheduler
ScheduleMain
{
// Has a new day dawned?
Do CheckForNewDay
// Are we within the current ad's showFor?
if ( ad.thisShowTime < ad.showFor )
{
// there is nothing to be done
return
}
// At this point, we know that we need a new ad
// Perform housekeeping tasks on the old one
do AdEndBookkeeping
// Pop out of a block if all ads on par
if ( block isn't all playlists )
{
find ad with minimum ad.numberShown
if ( ad.numberShown >= blockGoal )
set block to all playlists
}
// If we are over our quota of regular ads for the day,
// look for a runout
if ( adFaceTimeToday > faceTimeQuota )
{
do ShowARunout
}
else
{
Do ShowARegularAd
}
}
// end ad schedule main

```

FIG. 15A


```

////////////////////
// We must perform certain tasks when the calander day
// changes.
CheckForNewDay
{if ( the calander day has changed )
{
// Perform housekeeping tasks on the ad currently showing
Do StopShowingCurrentAd
// Runout ads are charged for a full showFor if they've been
// shown
// at all on a given day. Charge any runout ads if they've
// been
// shown at all.
for runout ads
{
if ( ad.thisShowTime > 0 )
{
ad.totalTimeShown += ad.showFor
ad.thisShowTime = 0
}
}
// Now, reset the counters for all ads to reflect the fact
// that
// a new day has dawned.
for all ads
{
ad.numberShownToday = 0
}
// Record yesterday's facetime
// Might not literally be yesterday, be sure to use
// whatever day the app was last run on
set old current day's facetime to totalFaceTimeToday
// and reset our global regular ad facetime counter
adFaceTimeToday = 0
totalFaceTimeToday = 0
// if we were in a block, back out
set block to all playlists
}
}
// end CheckForNewDay

```

FIG. 15B

```

////////////////////
// This function shows a runout ad, and if it
// can't find one, goes to a rerun
ShowARunout
{
for runout ads
{
// has the ad been flushed?
if ( ad.flushed )
try next ad
// are we done showing this runout today?
if ( ad.numberShownToday > ad.dayMax )
try next ad // this one's used up for the day
// are we done showing this runout for ever and ever?
if ( ad.shownFor > ad.shownForMax )
try next runout ad // this one's used up forever
// are we between the ad's start and end date?
if ( ad.startDate < the current date < ad.endDate )
try next runout ad
// the ad is not supposed to run today
// do we actually HAVE the ad?
if ( ad has not been downloaded )
{
ask for ad to be downloaded
try next ad
}
// ok, we believe we should show this runout
// we are now in runout state
Do ShowAnAd
return
}
// if we haven't found a runout ad, we will go to "rerun"
state
Do ShowARerun
}
// end ShowARunout

```

FIG. 15C

```

////////////////////
// Rerun state. Look for a regular ad to rerun
ShowARerun
{
for regular ads [ in current block ]
{
// has the ad been flushed?
if ( ad.flushed )
try next ad
// is this ad recent enough to rerun?
if ( ad.lastShownDate is older than returnInterval )
try next ad
// this one is too old to rerun
// if in block, show ads only if it's their "turn"
if ( ad.numberShownToday >= blockGoal )
try next ad // need to find a friend in this block
// are we between the ad's start and end dates?
if ( ad.startDate < the current date < ad.endDate )
try next ad
// the ad is not supposed to run today
// do we actually HAVE the ad?
if ( ad has not been downloaded )
{
ask for ad to be downloaded
try next ad
}
// ok, at this point we can show this ad, but because
// we're in rerun, we don't keep the books
Do ShowAnAd
return
}
// if we get here, we have no ads to show. Punt.
return
}
// end ShowARerun

```

FIG. 15D

```

////////////////////
// Show a regular ad
ShowARegularAd
{
for regular ads [ in current block ]
{
// has the ad been flushed?
if ( ad.flushed)
try next ad
// are we done showing this ad today?
if (ad.numberShownToday > ad.dayMax )
try next ad // this one's used up for the day
// if in block, show ads only if it's their "turn"
if ( ad.numberShownToday >= blockGoal )
try next add // need to find a friend in this block
// are we done showing this ad for ever and ever?
if ( ad.shownFor > ad.showForMax )
try next ad // this one's used up forever
// are we between the ad's start and end dates?
if ( ad.startDate < the current date < ad.endDate )
try next ad
// the ad is not supposed to run today
// do we actually HAVE the ad?
if ( ad has not been downloaded )
{
ask for ad to be downloaded
try next ad
}
// ok, we believe we should show this ad
// we are now in regular state
Do ShowAnAd
return
}
// if we get here, we have failed to find a regular
// ad. Go to runout
Do ShowARunout
}
// end ShowARegularAd

```

FIG. 15E

```

////////////////////////////////////
// Perform necessary housekeeping when we're taking
// down an ad
AdEndBookkeeping
{
// In rerun state, we don't do any bookkeeping
if ( in RerunState )
return
// Account for at most ad.showFor seconds, provided
// we've shown the ad for at least ad.showFor seconds
// Note that this means we don't charge for time beyond
// ad.showFor seconds, which is important
if ( ad.thisShowTime >= ad.showFor )
{
ad.numberShownToday += ad.showFor
ad.shownFor++
// we do NOT reset thisShowTime here, we do it in
// AdStartBookkeeping. It actually doesn't matter where
// we do it, provided we are careful NOT to do it for
// runout ads.
}
}
// end AdEndBookkeeping

```

FIG. 15F

```
////////////////////////////////////  
// Show an ad, including bookkeeping and block handling  
ShowAnAd  
{  
  // If the ad is in a block, notice that  
  if ( it's in a "block" playlist )  
  {  
    if ( not currently in a block )  
    {  
      find ad in block with minimum numberShown  
      make that our ad  
      set blockGoal to minimum numberShown+1  
    }  
    set current block to this playlist  
  }  
  // now do bookkeeping  
  Do AdStartBookkeeping  
  // and actually show it  
  Do DisplayThatAd  
}
```

FIG. 15G

```
////////////////////////////////////  
// Perform housekeeping when we put up an ad  
AdStartBookkeeping  
{  
// In return state, we don't do any bookkeeping  
if ( in RerunState )  
return  
// For regular ads  
if ( it's a regular ad )  
{  
ad.thisShowTime = 0  
ad.lastShownDate = now  
}  
}  
// end AdStartBookkeeping
```

FIG. 15H

PERSISTANT ADS	
PLAYLIST REQUEST	USED TO DETERMINE HOW MUCH ADVERTISING TO SEND TO CLIENT NOT USED
PLAYLIST RESPONSE CLIENTINFO	RELATIVELY LARGE: ONE OR MORE DAYS USED. SINGLE PLAYLIST COMPLETELY SPECIFIES LIST OF ADS CLIENT SHOULD HAVE
PLAYLIST RESPONSE SCHEDULING PARAMETERS	NOT USED

FIG. 16A

SHORT-LIVED ADS	
PLAYLIST REQUEST	NOT USED USED TO DETERMINE HOW MANY ADS CLIENT SHOULD RECEIVE
PLAYLIST RESPONSE CLIENTINFO	NOT USED. INSTEAD, CLIENT REQUEST NEW PLAYLIST WHENEVER ADS "RUN LOW". NOT USED
PLAYLIST RESPONSE SCHEDULING PARAMETERS	USED TO DETERMINE HOW LONG AN AD RUNS

FIG. 16B

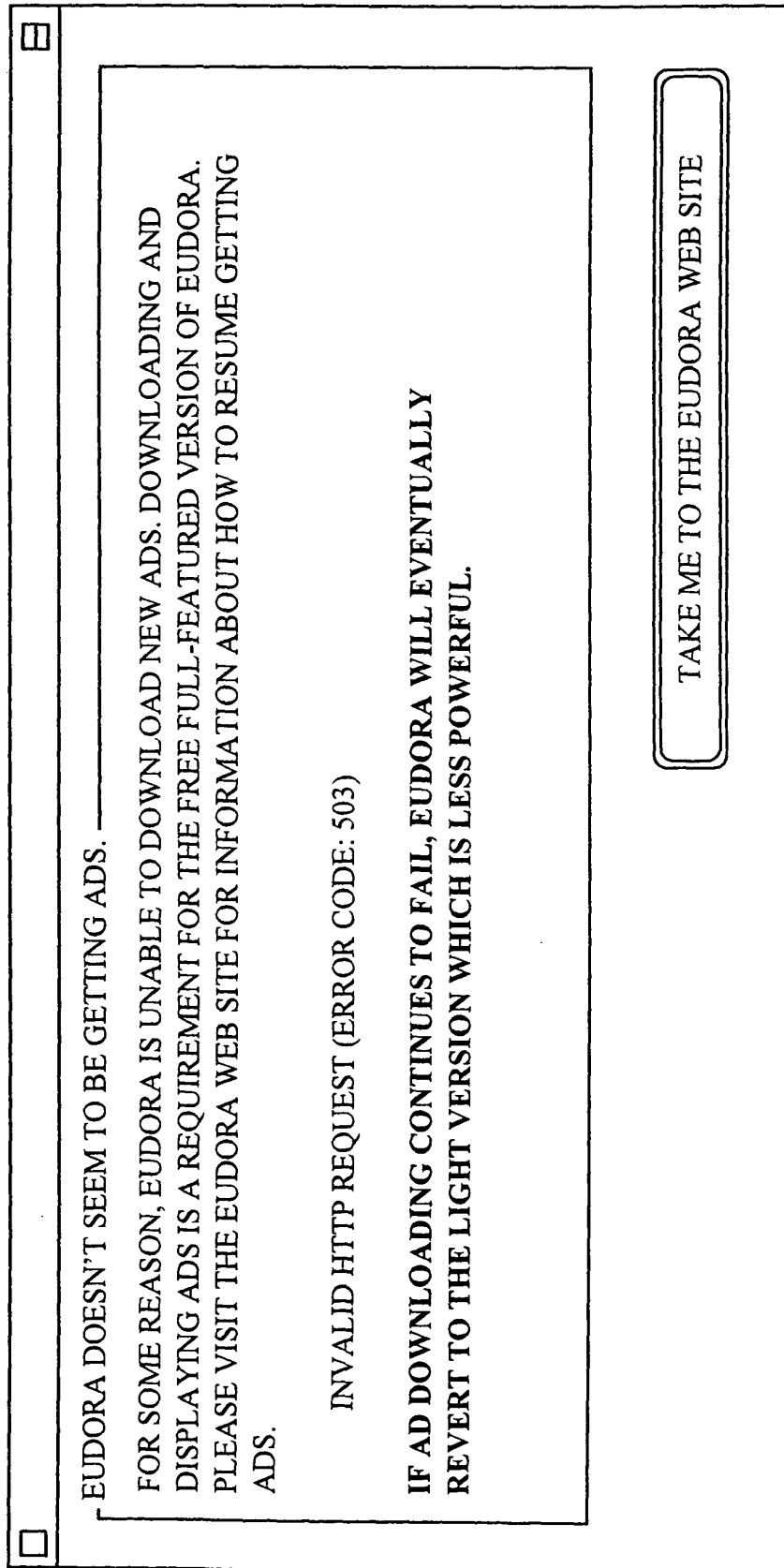


FIG. 17A

SOMETHING SEEMS TO BE COVERING THE AD.

ITS PROBABLY INADVERTENT, BUT EUDORA HAS DETERMINED THAT YOU ARE COVERING UP ALL OR A SIGNIFICANT PORTION OF AN AD. THE SOFTWARE IS DESIGNED TO NOTIFY YOU WHEN THIS HAPPENS IN THE HOPES THAT YOU WILL STOP COVERING UP THE AD. IF YOU DON'T, THIS WINDOW WILL KEEP POPPING UP (WHICH YOU WILL PROBABLY FIND QUITE ANNOYING).

WE'VE ALWAYS GOT SOME GOOD STUFF UNDER DEVELOPMENT BACK AT THE HOME OFFICE, AND IT'S THE ADVERTISING IN EUDORA THAT ENABLES US TO CONTINUE TO DEVELOP THE SOFTWARE WHILE PROVIDING IT TO YOU FOR FREE. WE'VE WORKED HARD TO MAKE SURE THE ADVERTISING ISN'T ANNOYING AND WE GENUINELY HOPE THAT YOU ARE NOT DELIBERATELY TRYING TO COVER THE ADS BECAUSE THEY'RE BOTHERING YOU. OF COURSE, YOU CAN CHOOSE TO PAY US FOR EUDORA BY CHOOSING "PAYMENT & REGISTRATION" FROM THE "HELP" MENU AND CLICKING ON "PAID FULL VERSION." OR YOU CAN REMOVE WHATEVER IS OBSCURING THE AD.

OK

FIG. 17B

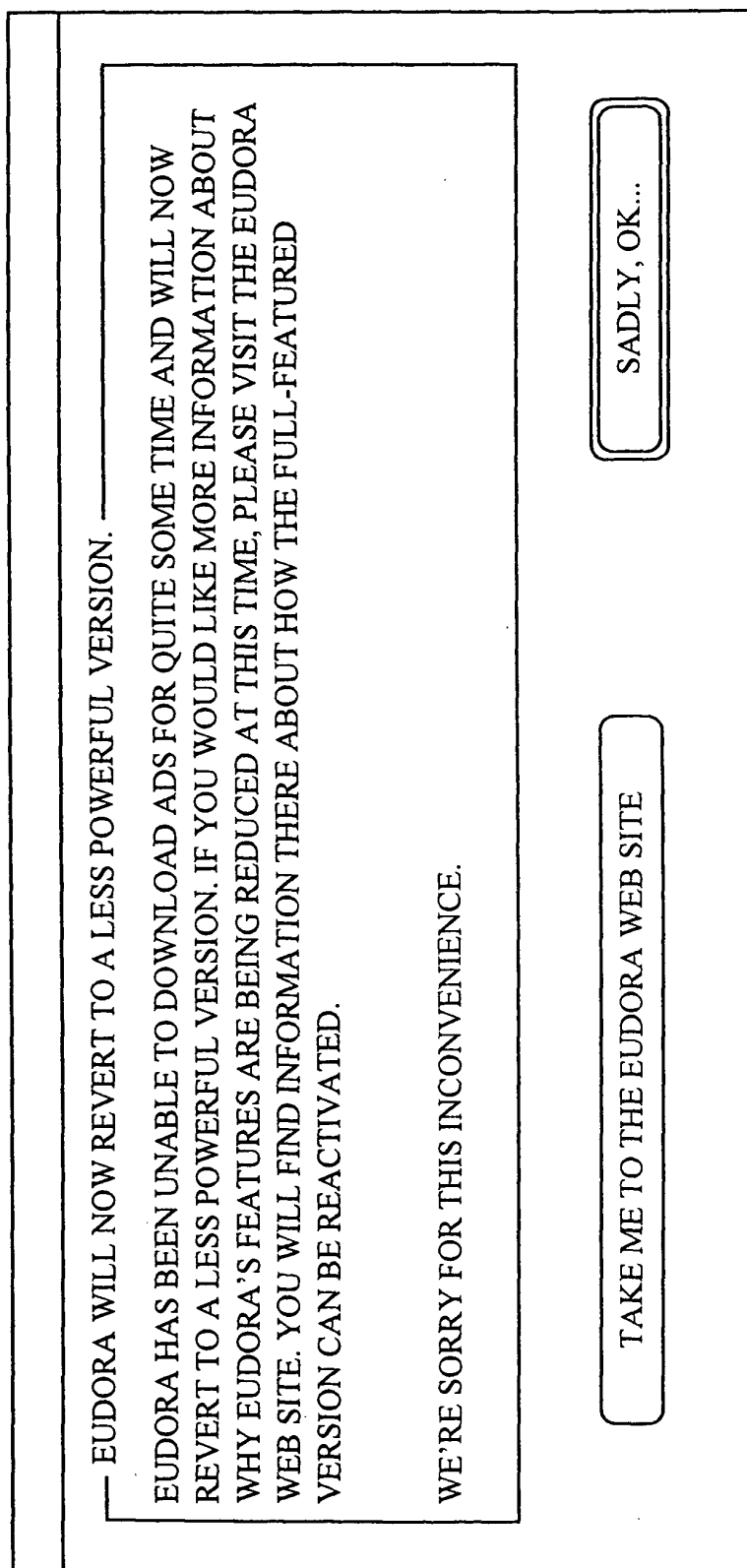


FIG. 17C

FIG. 18A

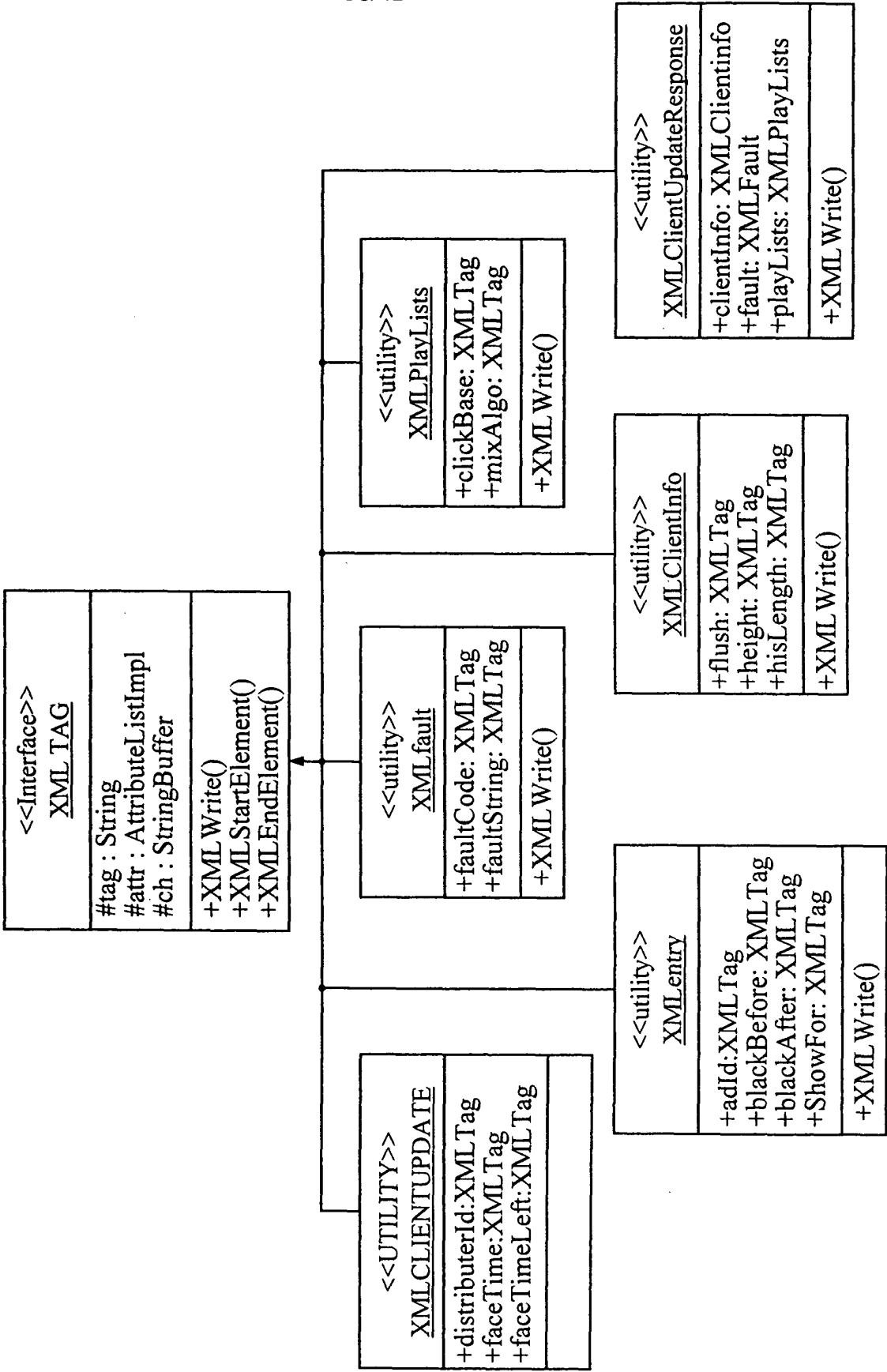


FIG. 20

% The list of available ads advantageously can be built from the following query:

```
ads = dbCon.prepareStatement("SELECT" * FROM ads WHERE StartDate<=today AND endDate>=today + 30 AND
AdType = "I" AND AdStatus = "A" AND ImpressionsServed<Impressions ORDERED BY ImpressionsServed ASC);

run out ads = dbCon.prepareStatement("SELECT * FROM ads WHERE StartDate <= today AND endDate >= today +
30 AND AdType = "R" AND AdStatus = "A" AND ImpressionsServed<Impressions ORDERED BY ImpressionsServed
ASC);
```

% The time required to deliver the ads advantageously can be calculated in the following manner.

face time left for today [seconds] = faceTime[today] - faceTimeUsedToday

(Comment: Face time left for today is the number of seconds the servlet can use to deliver special ads today.)

predict face time [seconds] = SUM(faceTime[tomorrow], faceTime[tomorrow+1], ... faceTime[tomorrow+reqInterval])

(Comment: Predict face time is the number of seconds the servlet predicts the user is going to have.)

goal show time left [seconds] = predict face time - faceTimeLeft

(Comment: Goal show time left is the number of seconds that the software provider needs to fill with ads.)

FIG. 21A

```

% Targeting
while (face time left for today) {
    if ad is not in the history {
        select ad [according to target = today]
        face time left for today -= ad.showFor
    }
    next ad
}

while (Goal show time left ) {
    if ad is not in the history {
        select ad [according to target]
        goal show time left -= ad.showFor
    }
    next ad
}

```

Default values:

```

reqInterval = 1 day.
facetime = 30 minutes
faceTimeQuota is ?
histLength = 31 days

```

FIG. 21B

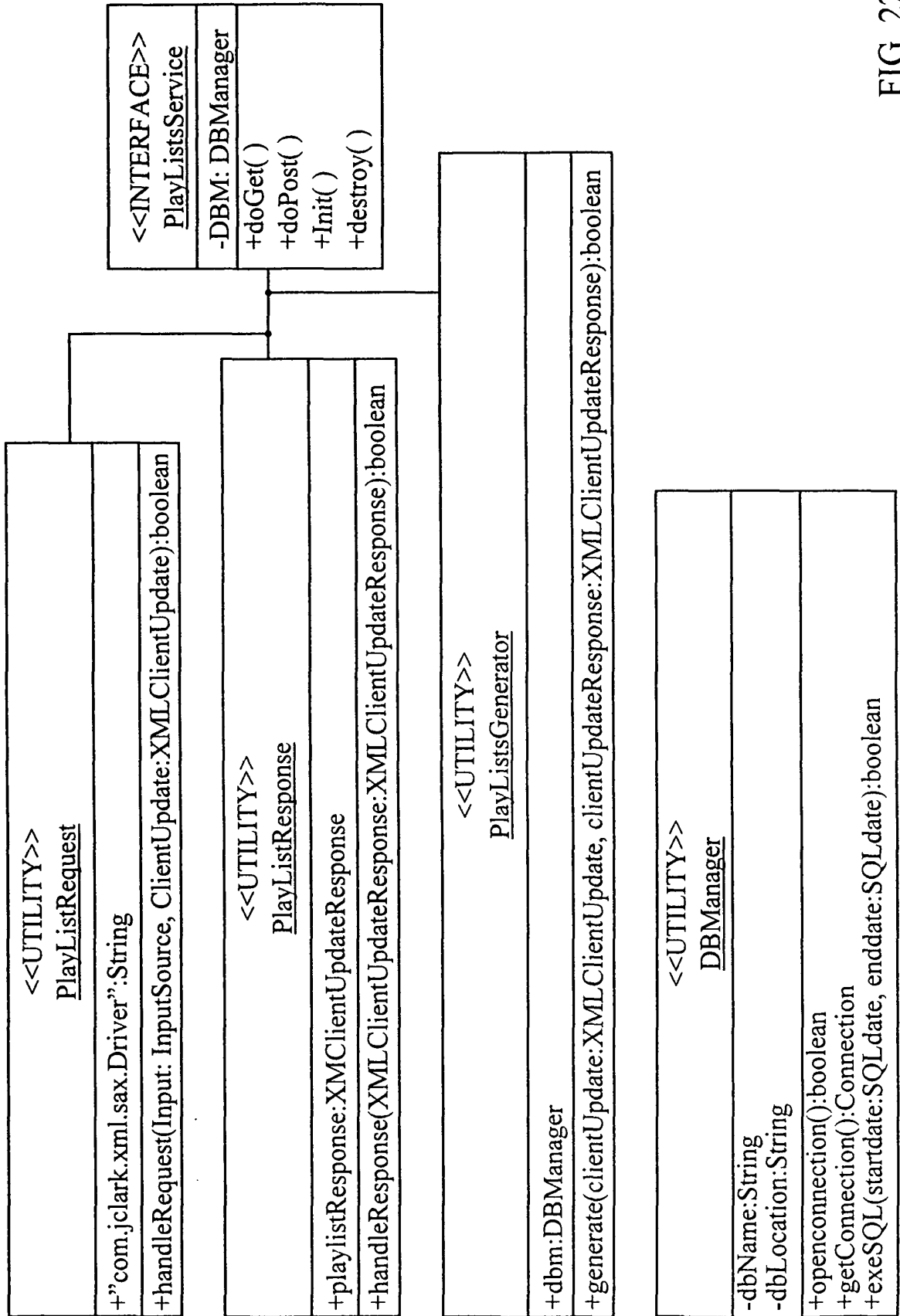


FIG. 22

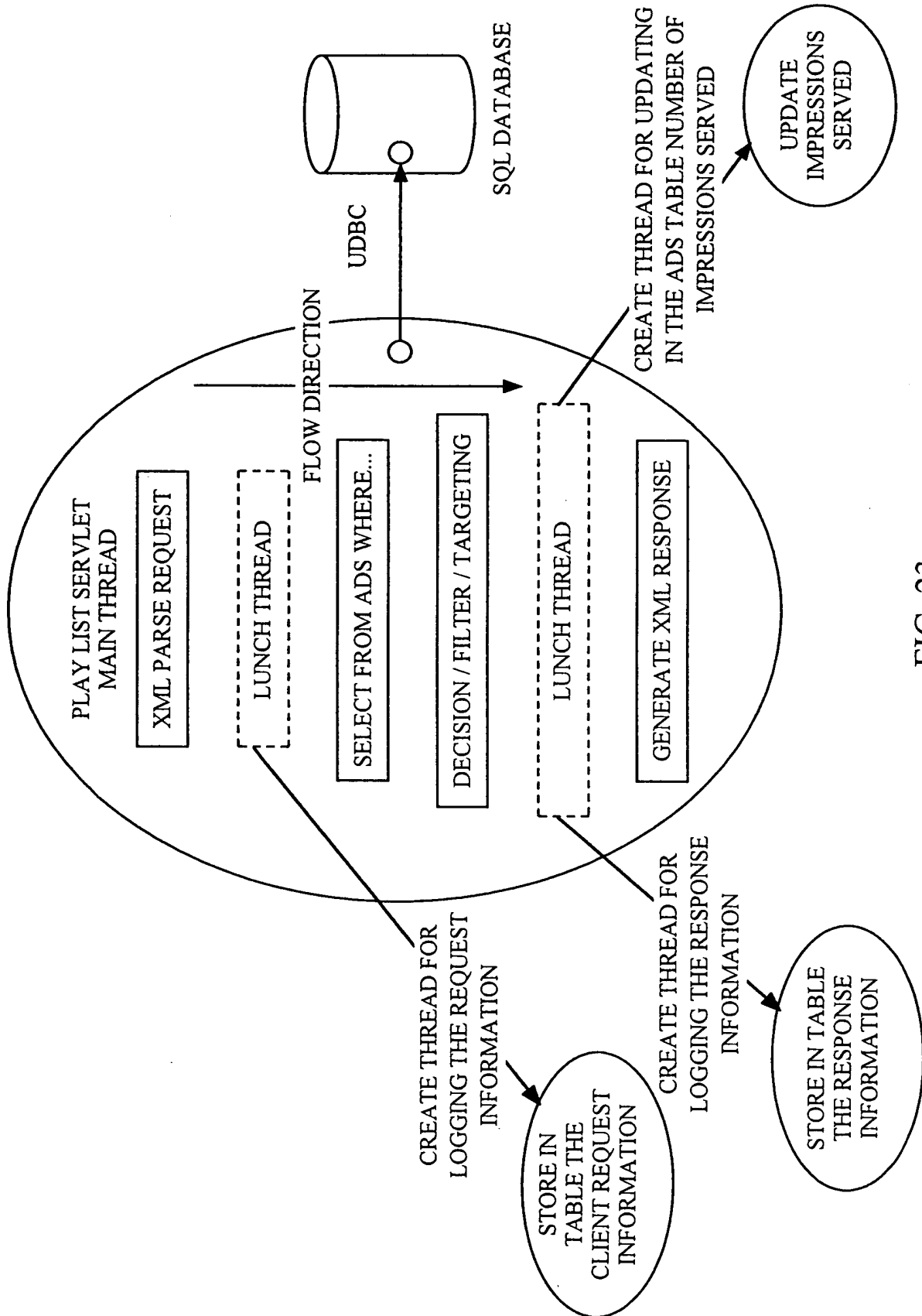


FIG. 23